Big data management with NoSQL and NewSQL

Me

- Olivier Curé
  - Associate Professor LIGM
  - Research domains: Big data, Artificial Intelligence, Semantic Web
  - Coordinates
    - olivier.cure@univ-eiffel.fr
    - Office 4B130 Copernic
    - Twitter: @oliviercure

Four courses

1) **Big data batch processing**
2) **Big data management with NoSQL and NewSQL**
3) **Symbolic Artificial Intelligence and the Semantic Web**
4) **Big data stream processing**

- For courses #1, #2, #3: 6 lectures + 6 lab sessions
- For course #4: 4 lectures and 4 lab sessions
- Grades:
  - Final exam + Quizzes + Involvement in labs and courses

Overview of the 4 courses

1) Processing big data.
   - Batch processing of unstructured to structured data
   - MapReduce programming model
2) Storing/querying big data
   - New data models
   - New query languages
3) Knowledge representation on the Web
   - Graph-based
   - Reasoning
4) Streaming processing and pub/sub systems
   - Streaming enabled DBMS
   - Big data stream processing engines

This course

- Intro to Big data
- 4 families of NoSQL (Not only SQL) databases
- NewSQL
- Polyglot persistence

Data is the new oil

https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data

Data is the new oil

Just like oil needs to be refined to produce goods like gas, plastic, chemicals, etc. Data must be transformed, and analyzed for it to be of value.

Big data

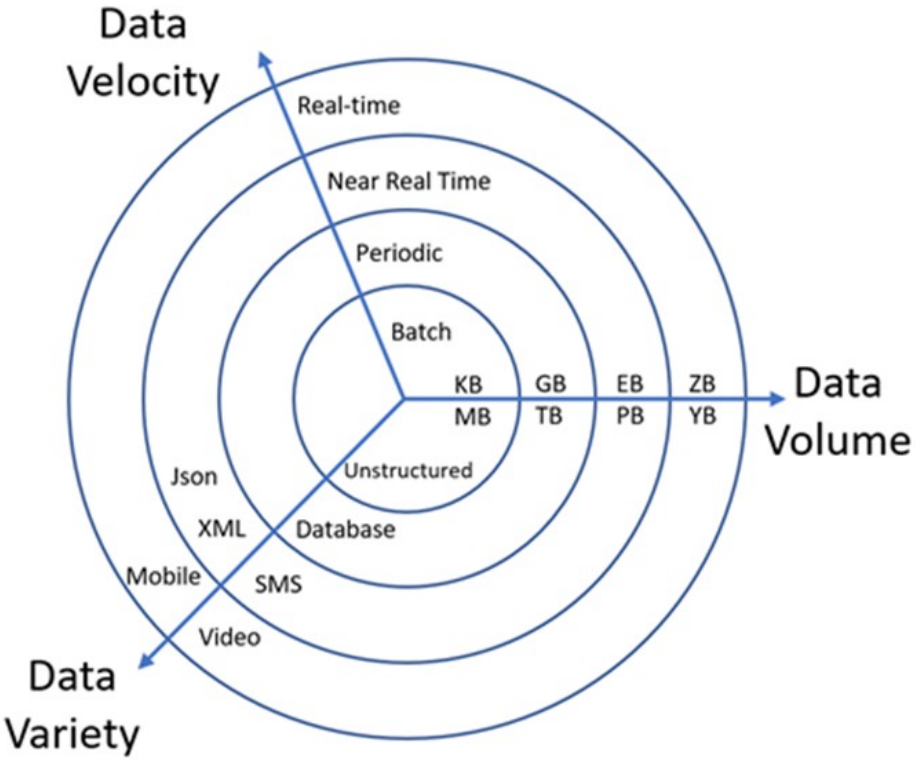

BIG
DATA
STORAGE
ANALYTICS
TECHNOLOGIES
SOFTWARE
INTERNET
SIZE
NEEDED
PETABYTES
MANAGEMENT
INFORMATION
SEARCH
INCLUDE
SYSTEMS
NETWORKS
RECORDS
SHARED
DISK
TIME
HUNDREDS
COMPLEX
DATABASES
RESEARCH
SETS
EXAMPLES
ABILITY
SOCIAL
LARGE
EVERY
DISTRIBUTED
PARALLEL
SAN
TOLERABLE
USE
ONE
USING
GARTNER
MASSIVELY
BIOLOGICAL
WITHIN
PROCESS
DEFINITION
TOOLS
SET
GENOMICS
PERFORMANCE
ALSO
RELATIONAL
TENS
CAPACITY
PRACTITIONERS
BUSINESS
MAY
CAPTURE
MANAGE
GROW
LOGS
PROCESSING
COST
CONTINUES
CITATION
TERABYTES
DIFFICULTY
TARGET
SENSOR
ARCHIVES
APPLIED
CURRENT
CASE
LARGER

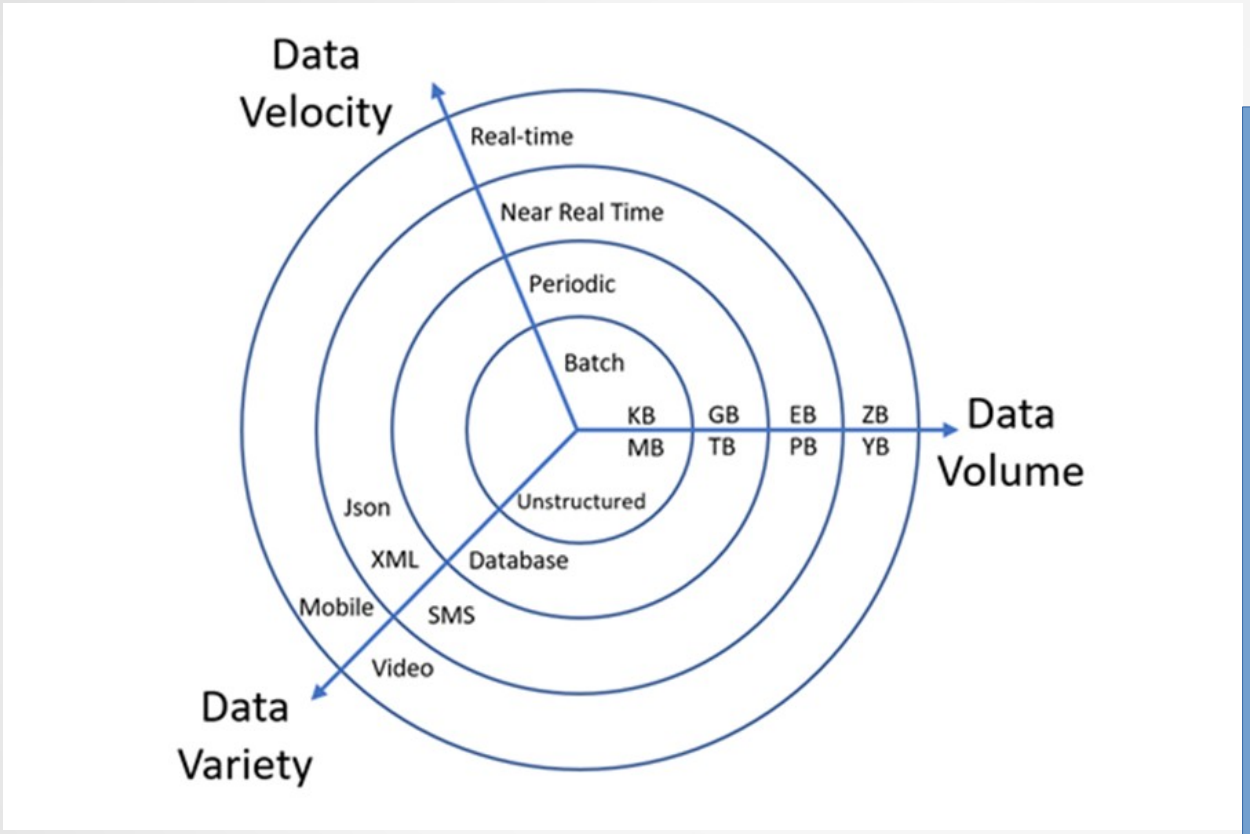
Big data

- Wikipedia definition
  - Big data usually includes data sets with sizes **beyond the ability of commonly-used software** tools to capture, curate, manage, and process the data within a tolerable elapsed time.
- It refers to technologies and initiatives involving data that is too diverse, fast-changing and massive for conventional technologies, skills and infrastructure to address efficiently.
- So the **Volume**, **Velocity** and **Variety** of data is too great.
- All terms start with a 'V', hence the 3Vs of Big data.

From bit to YottaByte

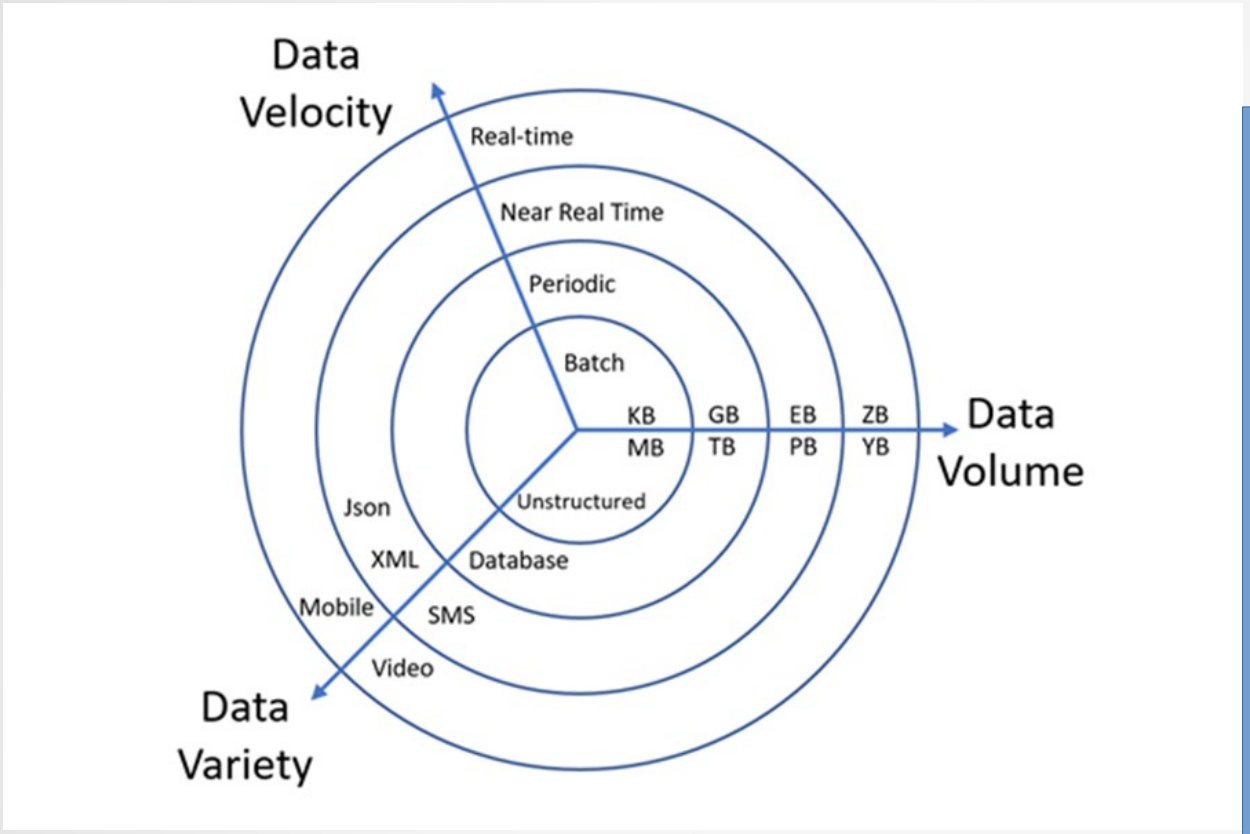
Abbreviation	Unit	Value	Size (in bytes)
b	bit	0 or 1	1/8 of a byte
B	bytes	8 bits	1 byte
KB	kilobytes	1,000 bytes	1,000 bytes
MB	megabyte	$1{,}000^2$ bytes	1,000,000 bytes
GB	gigabyte	$1{,}000^3$ bytes	1,000,000,000 bytes
TB	terabyte	$1{,}000^4$ bytes	1,000,000,000,000 bytes
PB	petabyte	$1{,}000^5$ bytes	1,000,000,000,000,000 bytes
EB	exabyte	$1{,}000^6$ bytes	1,000,000,000,000,000,000 bytes
ZB	zettabyte	$1{,}000^7$ bytes	1,000,000,000,000,000,000,000 bytes
YB	yottabyte	$1{,}000^8$ bytes	1,000,000,000,000,000,000,000,000 bytes


Data
Velocity
Real-time
Near Real Time
Periodic
Batch
KB
MB
GB
TB
EB
PB
ZB
YB
Data
Volume
Unstructured
Json
XML
Database
Mobile
SMS
Video
Data
Variety



Data volumes:

- 10+ TB:
  - A high end server
- 10+ PB
  - US Library of Congress
- 100+ PB
  - DropBox (2016)
  - Netflix (2017)
  - Twitter (2017)
- 1+ EB
  - Amazon, Google, Facebook
- 1 ZB
  - Global internet traffic in 2019

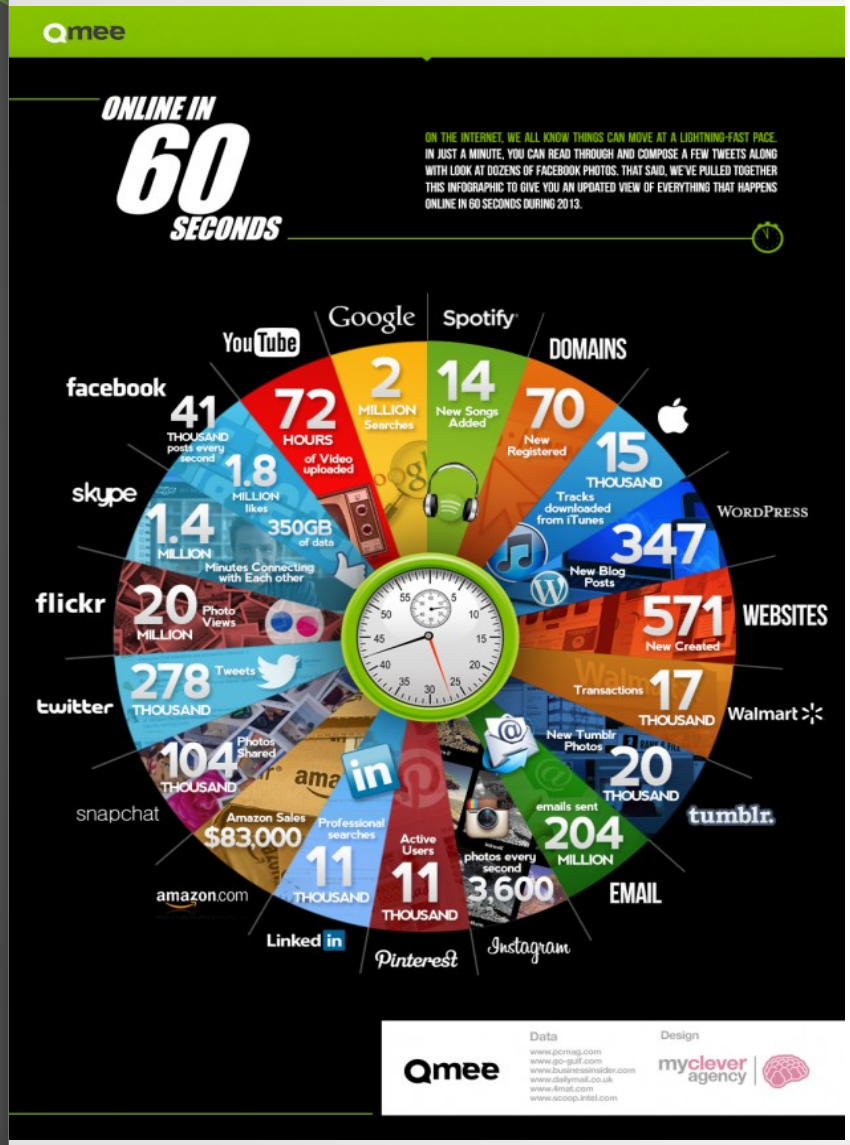


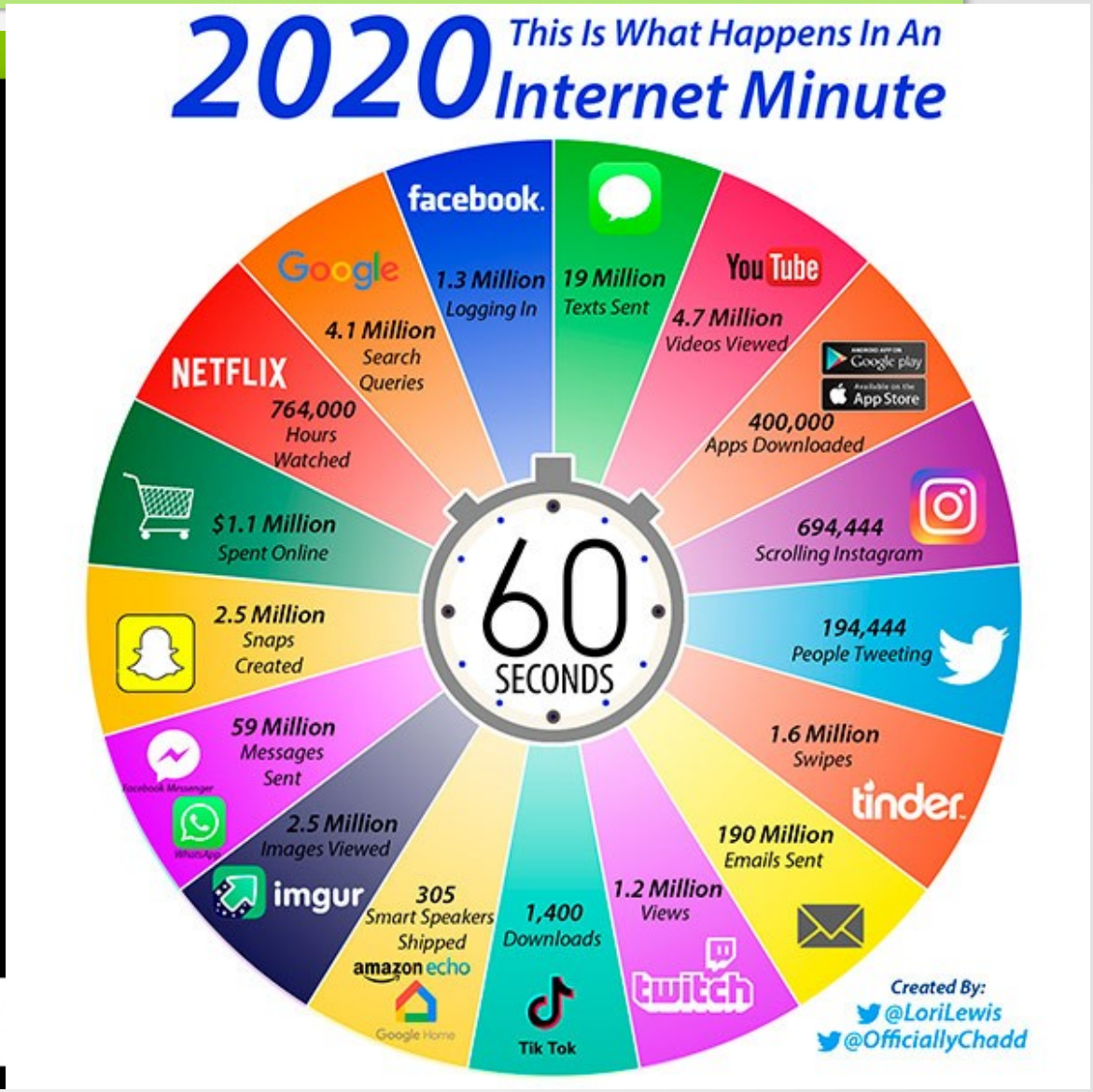
- More Vs:
  - Veracity
  - Vocabulary
  - Venue

Data volumes:

- 10+ TB:
  - A high end server
- 10+ PB
  - US Library of Congress
- 100+ PB
  - DropBox (2016)
  - Netflix (2017)
  - Twitter (2017)
- 1+ EB
  - Amazon, Google, Facebook
- 1 ZB
  - Global internet traffic in 2019

Online in 60 sec (from 2013 to 2020)





Why Big data

- Why big data
  - The Web (2.0) and availability of (open) data
  - Increase of storage capacities
  - Increase of processing power

Two forms of Big volumes

- ’small analytics’: SQL on very large data sets. Using aggregate ops of SQL.

- ’big analytics’: Data clustering, regressions, machine learning. Using statistical tools: R, SPSS (Statistical Product and Service Solutions - IBM), SAS.

Making sense at scale

- Machines: cloud computing
- Algorithms: machine learning and analytics (Etienne Côme's course)
- People: crowdsourcing and human computation

EXPERT OPINION

Contact Editor: **Brian Brannon**, bbrannon@computer.org

The Unreasonable Effectiveness of Data

Alon Halevy, Peter Norvig, and Fernando Pereira, *Google*

Eugene Wigner's article "The Unreasonable Effectiveness of Mathematics in the Natural Sciences"[1] examines why so much of physics can be neatly explained with simple mathematical formulas such as $f = ma$ or $e = mc^2$. Meanwhile, sciences that involve human beings rather than elementary particles have proven more resistant to elegant mathematics. Economists suffer from physics envy over their inability to neatly model human behavior. An informal, incomplete grammar of the English language runs over 1,700 pages.[2] Perhaps when it comes to natural language processing and related fields, we're doomed to complex theories that will never have the elegance of physics equations. But if that's so, we should stop acting as if our goal is to author extremely elegant theories, and instead embrace complexity and make use of the best ally we have: the unreasonable effectiveness of data.

One of us, as an undergraduate at Brown University, remembers the excitement of having access to the Brown Corpus, containing one million English words.[3] Since then, our field has seen several notable corpora that are about 100 times larger, and in 2006, Google released a trillion-word corpus with frequency counts for all sequences up to five words long.[4] In some ways this corpus is a step backwards from the Brown Corpus: it's taken from unfiltered Web pages and thus contains incomplete sentences, spelling errors, grammatical errors, and all sorts of other errors. It's not annotated with carefully hand-corrected part-of-speech tags. But the fact that it's a million times larger than the Brown Corpus outweighs these drawbacks. A trillion-word corpus—along with other Web-derived corpora of millions, billions, or trillions of links, videos, images, tables, and user interactions—captures even very rare aspects of human behavior. So, this corpus could serve as the basis of a complete model for certain tasks—if only we knew how to extract the model from the data.

Learning from Text at Web Scale

The biggest successes in natural-language-related machine learning have been statistical speech recognition and statistical machine translation. The reason for these successes is not that these tasks are easier than other tasks; they are in fact much harder than tasks such as document classification that extract just a few bits of information from each document. The reason is that translation is a natural task routinely done every day for a real human need (think of the operations of the European Union or of news agencies). The same is true of speech transcription (think of closed-caption broadcasts). In other words, a large training set of the input-output behavior that we seek to automate is available to us *in the wild*. In contrast, traditional natural language processing problems such as document classification, part-of-speech tagging, named-entity recognition, or parsing are not routine tasks, so they have no large corpus available in the wild. Instead, a corpus for these tasks requires skilled human annotation. Such annotation is not only slow and expensive to acquire but also difficult for experts to agree on, being bedeviled by many of the difficulties we discuss later in relation to the Semantic Web. The first lesson of Web-scale learning is to use available large-scale data rather than hoping for annotated data that isn't available. For instance, we find that useful semantic relationships can be automatically learned from the statistics of search queries and the corresponding results[5] or from the accumulated evidence of Web-based text patterns and formatted tables,[6] in both cases without needing any manually annotated data.

1541-1672/09/$25.00 © 2009 IEEE
Published by the IEEE Computer Society

Crowdsourcing and Human computation

- **Crowdsourcing** first coined in 2006 in Wired magazine: ’a task of taking a job traditionally performed by a designated agent and outsourcing it to an undefined, generally large group of people in the form of an open call’.
- **Human computation** (van Hahn , 05) : ’... a paradigm for utilizing human processing power to solve problems that computers cannot yet solve’.
- **Global Brain** (Bernstein, CACM15) : people and computers to constitute a global brain. Ask for new programming metaphors to program it.

Machines

- Two main solutions to process big data: MapReduce-based frameworks and high compression approaches.

- Using them together will be more and more frequent in future systems.

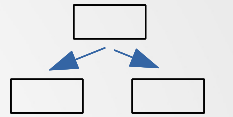
- For instance by distributing highly compressed data over a cluster of machines

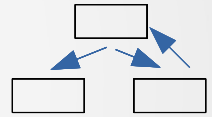
Some DBMS history

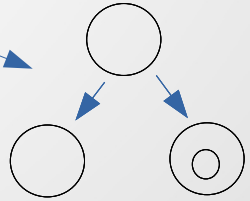
- DBMS: Database management system
- Data model: A data model organizes data elements and standardizes how the data elements relate to one another (Wikipedia).
- Usually takes care of:
  - Data types
  - Constraints
  - Operations
  - Semantics

Some history (2)

- Flat model (50s)
- Hierarchical model (60s)
- Network model (70s)
- Relational model (80s)
- Object oriented model (90s)
- Object-relational model (90s)
- Semi-structured, XML (2000s)
- ...







Flat model

- Until the 60s
- Problems:
  - Data redundancy
  - At update-time
  - No abstract data model
  - Requires some knowledge on the physical storage organization
  - No common query language

Hierarchical model

- Tree-like model
- Single upward link in each record
- Record are sorted
- Problems
  - Modifying data structures imposes to change programs
  - Programmers need some time to get used to the database strcuture.
- Systems: North American Rockwell & IBM : IMS (Information Management System)

Network model

- An extension of the hierarchical model
- Standardization in 1971 by the CODASYL group (Conference on Data Systems languages)
- Two fundamental constructs: records and sets.
- Records contain fields
- Sets define one-to-many relationships between records: one owner, many members.
- Problems: navigation in the DB is harder than in hierarchical model.
- Systems: IDMS (Integrated Database Management System)

Relational model

- Proposed by E.F. Codd (IBM) in early 70s.
- Relations are connected (primary key, foreign key)
- Declarative query language : SQL → optimization
  - Set based operations (relational algebra)
- 1st Systems: System R, Ingres (INteractive Graphics REtrieval System), Oracle

Information Retrieval P. BAXENDALE, Editor

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. Introduction

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for noninferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

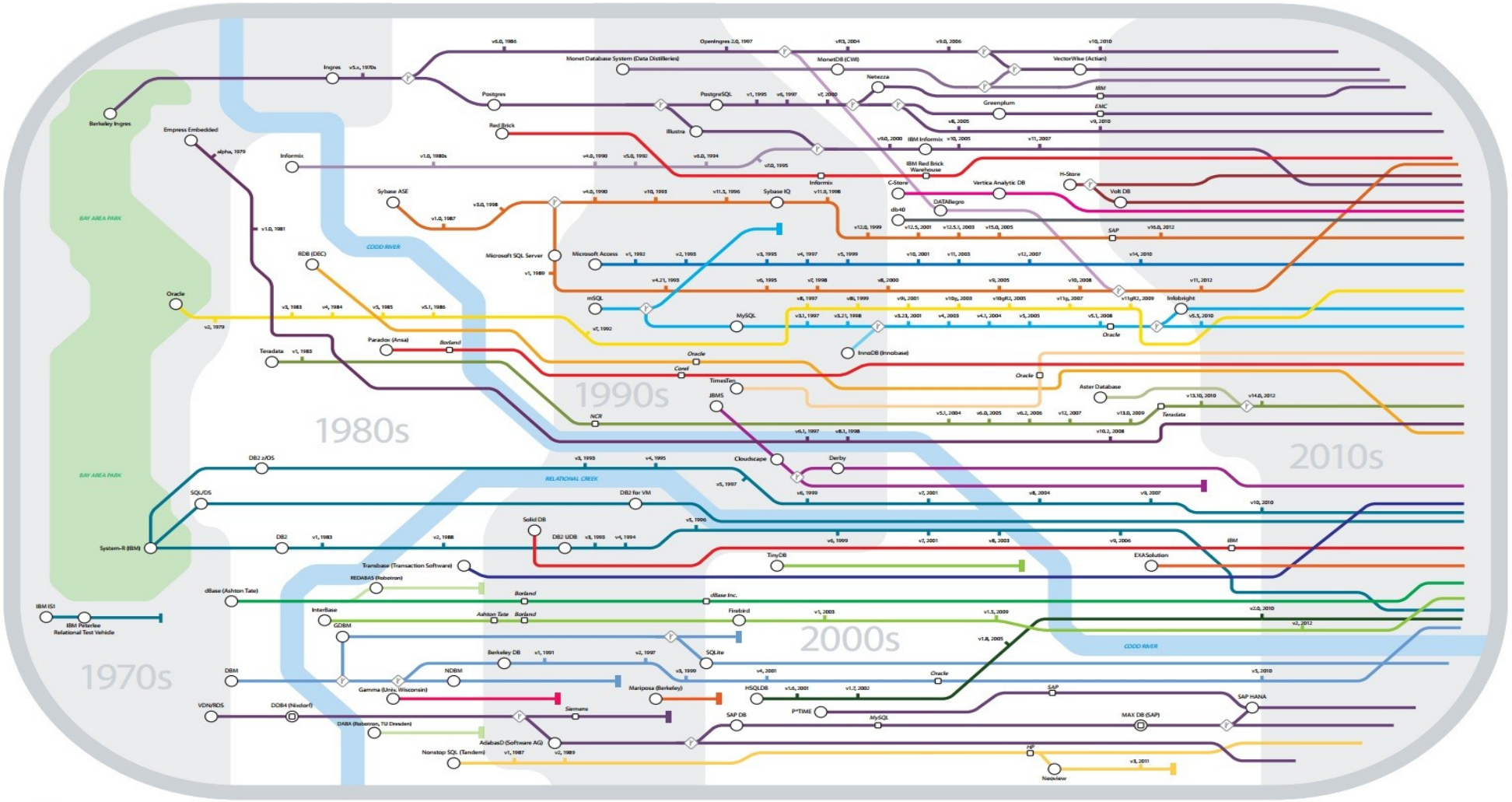
Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. Data Dependencies in Present Systems

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.2.1. *Ordering Dependence.* Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

RDBMS genealogy



Other models

- Object oriented
  - To cope with the impedance mismatch problem encounter with OOP
    - Impedance mismatch: the difference between the relational model and the in-memory data structures
    - The values in a RDBMS have to be simple (first normal form): no list, no nested records. This is not true for in-memory data structures.
    - Dealing with impedance mismatch using Object Relational Mapping (ORM, like Hibernate)
    - Emergence of object relational DBMS

Other models

- XML
  - Trees
  - Query languages: Xpath, Xquery
  - Native XML Database systems: Mark Logic, eXist DB, Berkeley DB
  - Emergence of XML support in ORDBMS

Pros of RDBMS

- Native support of concurrency control via ACID transactions
- Well identified model
- SQL standard

- In general shared database integration
  - Several applications tackling one database
- In opposition to application database
  - One database per application

Limitations of RDBMS

- RDBMS were not designed to run on clusters
  - Oracle RAC and SQL Server work on the concept of shared disk:
    - They use a cluster-aware file system that writes to a highly available disk subsystem, which is Single Point of Failure. This has scaling limitations
  - Commercial RDBMS are usually priced on a single-server assumption, so running on a cluster rapidly becomes very expensive

Database ecosystem

Database ecosystem

Relational

MySQL

ScaleDB

Object Relational

SYBASE

MariaDB

Column stores / data warehouse

snowflake

Greenplum

SAP HANA

vectorwise

Google BigQuery

PARACCEL

aster data
big data. fast insights.

NETEZZA

INFOBRIGHT

DATAllegro
Microsoft

teradata.

Amazon Redshift

XTREMEDATA

Embedded

SQLite

Apache Derby

HyperSQL

Memory centric

VoltDB

solidDB

Oracle TenTimes
SAP BI Accelerator

Object

VERSANT
THE DATABASE FOR DEVELOPERS

Objectivity
Innovate with Confidence

db4objects

GEMSTONE

XML

Tamino

eXist

xindice

ORACLE
BERKELEY DB

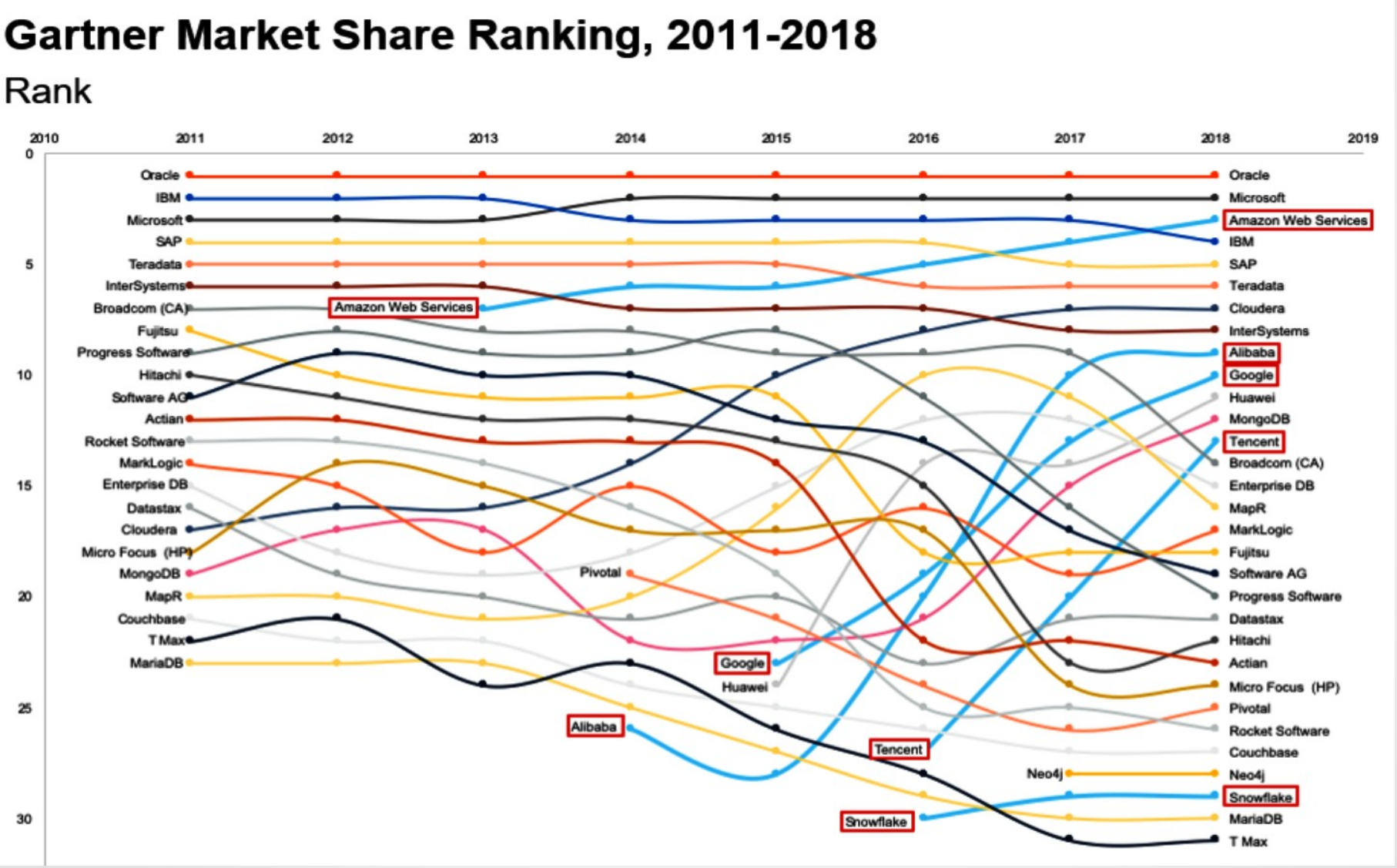
Network

IDMS

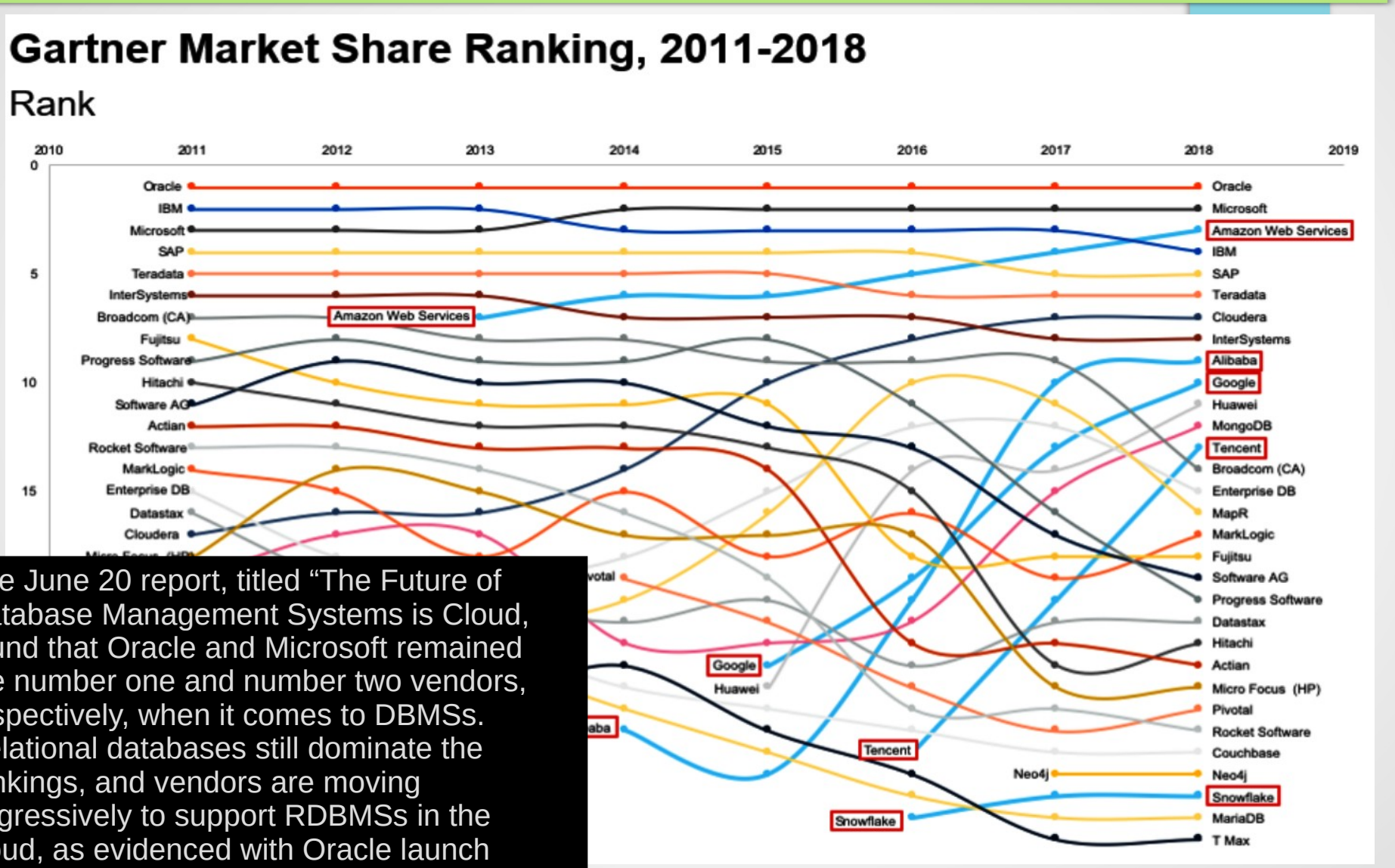
Hierarchy

IBM IMS

RDBMS market



RDBMS market



The June 20 report, titled “The Future of Database Management Systems is Cloud, found that Oracle and Microsoft remained the number one and number two vendors, respectively, when it comes to DBMSs. Relational databases still dominate the rankings, and vendors are moving aggressively to support RDBMSs in the cloud, as evidenced with Oracle launch today of Autonomous Database Dedicated, a new private database service that’s available in the Oracle public cloud.

NoSQL (Not only SQL)

- No real definition of NoSQL
- NoSQL is more a movement than a technology
- A tentative definition by Forrester Research Inc.

  “ a no-relational database management system that provides storage, processing, and accessing of complex data structures and support for large volumes of poly-structured data. It supports a horizontal, on-demand, extreme scale-out database platform tat delivers a schema-less and flexible data model, and is optimized for high performance.”

NoSQL

- Consider NoSQL:
  - To support complex business apps requiring transactional and operational databases
  - To process data faster by scaling horizontally across a clustered configurations
  - To manage and store increasing volumes of structured and unstructured data
  - To support low latency ad hoc queries
  - To support next generation business apps, e.g., Web 2 and 3, mobile, cloud.

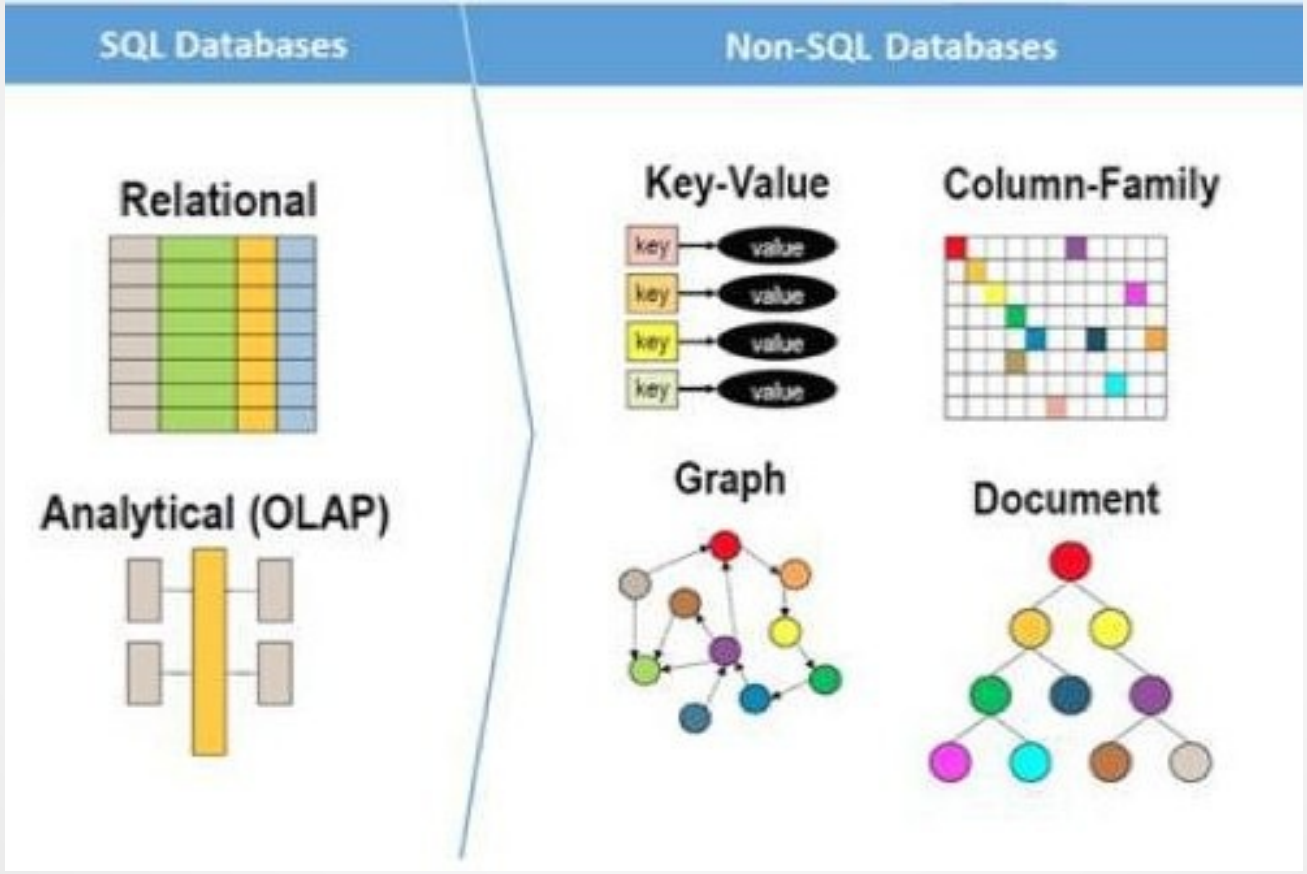
NoSQL

- Two reasons to consider NoSQL:
  - to handle data access with sizes and performance that demand a cluster
  - to improve the productivity of application development by using a more convenient data interaction style (limit the impact of the impedance mismatch)

NoSQL

- Common characteristics of NoSQL (at least before some evolutions):
  - Not using the relational model
  - Running well on clusters (except Graph stores)
  - Open-source
  - Schemaless (schema on read vs Schema on write)
  - Rarely a declarative query language, more an API based approach

Types of NoSQL stores



NoSQL ecosystem

Key Value stores

Voldemort
TokyoCabinet
Scalaris
Amazon Dynamo & SimpleDB

Dynomite

Graph databases

Document databases

Terrastore

Column family databases

Google BigTable

Polyglot persistence

- Term coined after Neal Ford's Polygot programming, asking to write programs with a mix of prog. languages.
- Polyglot persistence aims to use different data stores in your applications.
- Imagine an e-commerce application. What would you use for the shopping cart, the completed orders, session data, product inventory, etc. ?